

Index

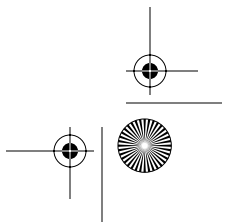
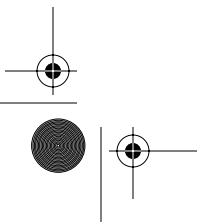
A

Absorbing class, 117
Abstract Factory, 70–71
Accept methods, 327
Accumulation methods, 315,
325–330
Accumulation refactorings
Collecting Parameter pattern, 30,
313–319
Move Accumulation to Collecting
Parameter, 313–319
Move Accumulation to Visitor,
320–338
Parameter pattern, 313–319
Visitor pattern, 31, 320–338
Active nothing, 302
Adapter classes, 249
Adapter pattern
See also Alternative Classes with
Different Interfaces smell
See also Duplicated Code smell
See also Oddball Solution smell
Adapting with Anonymous Inner
Classes, 258–268
description, 247–257
extracting, 258–268
patterns-based refactoring, 30
vs. Facade pattern, 259–260

Adapting with Anonymous Inner
Classes, 258–268
Alexander, Christopher, 23–24, 26
Alternative Classes with Different
Interfaces smell, 43. *See also*
Unify Interfaces with Adapter.
Anderson, Bruce, 302
APIs, supporting multiple versions,
258–268
Arnoldi, Massimo, 116

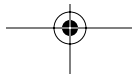
B

Barzun, Jacques, 12, 37
BDUF (big design up-front), 34–35
Beck, Kent
acknowledgments, xxi, xxiii, xxv
on Composed Method, 123
continuous refactoring, 5
hard-coded notifications, 237
identifying smells, 37
information accumulation, 314
JUnit, 33
pattern-directed refactoring, 29
“red, green, refactor,” 5
on Singletons, 116
Smalltalk Best Practices Patterns,
313
TDD (test-driven development), 5





- Big design up-front (BDUF), 34–35
- Bloch, Joshua, 59
- Books and publications
 - Checks: A Pattern Language...*, 24
 - Contributing to Eclipse*, 24
 - Design Patterns*, xxi, xxiii, 24
 - Domain-Driven Design*, xxiv, 24
 - Extreme Programming Explained*, 24
 - A Pattern Language*, 23
 - “Patterns & XP,” xxii
 - Patterns of Enterprise Application Architectures*, 24
 - Refactoring*, xx, xxi
 - Simple and Direct*, 12
 - Smalltalk Best Practices Patterns*, 313
 - Test-Driven Development*, 6
 - Test-Driven Development: A Practical Guide*, 6
 - A Timeless Way of Building*, 23
 - UML Distilled*, xxi
- Builder pattern. *See also* Primitive Obsession smell.
 - description, 96–113
 - encapsulating Composites, 96–113
 - patterns-based refactoring, 30
 - performance improvement, 109–111
 - Schema-Based Builder, 111–113
- C
- Cascading notifications, 237
- Catalogs of patterns. *See also specific patterns; specific refactorings.*
 - Design Patterns*, 24
 - Example section, 49
 - HTML Parser, 50–51
 - loan risk calculator, 51
 - Mechanics section, 48–49
 - Motivation section, 48
 - Name section, 47
 - refactoring formats, 47–49
 - sketches, 47
 - Summary section, 47–48
 - Variations section, 49
 - XML builders, 50
- Catch-all constructors, 341
- Chain Constructors, 340–342. *See also* Duplicated Code smell.
- Chaining constructors, 340–342
- Checks: A Pattern Language...*, 24
- Child containers, 216
- Clarifying unclear code. *See* Simplification refactorings.
- Class pattern, 286–295. *See also* Primitive Obsession smell.
- Classes. *See also* Subclasses.
 - absorbing, 117
 - adapter, 249
 - child containers, 216
 - composite, 215
 - context, 168
 - core responsibility embellishments, 144–165
 - delegating, 347
 - direct instantiation, 80–87
 - double-dispatch, 321
 - embellished, refactoring, 145–165
 - encapsulating, 80–87
 - excessive instances, refactoring, 43
 - heterogeneous, 320–338
 - information accumulation, 320–338
 - interfaces, refactoring, 43
 - interpreting simple languages, 269–284
 - multiple instance, 298
 - new adapter, 261



- nonterminal expressions, 270
- notifiers, 238
- one/many distinctions, 224–235
- overburdened adapters, 261
- polymorphic creation, 88–95
- receivers, 238
- Singletons, refactoring, 43–44
- state superclass, 168
- subject superclasses, 238
- terminal expressions, 270
- unifying interfaces, 247–257, 343–345
- visibility, refactoring, 42–43
- visitee, 327
- Code
 - accumulated clutter, 15–16
 - clarifying. *See* Refactoring; Simplification refactorings; *specific refactorings.*
 - continuous refactoring, 5, 13–14
 - design debt, 15–16
 - duplicate. *See* Duplicated Code smell.
 - generalizing. *See* Generalization refactorings.
 - human-readable, 12–13
 - implicit languages, 271
 - interpreting simple languages, 269–284
 - nonterminal expressions, 270
 - patterns and complexity, 31–32
 - premature optimization, 296–297
 - problem indicators. *See* smells.
 - protecting. *See* Protection refactorings.
 - simplifying. *See* Refactoring; Simplification refactorings; *specific refactorings.*
 - sprawl, refactoring, 43, 68–79
 - terminal expressions, 270
 - type safety, 286–295
 - uncalled, refactoring, 58
- Code smells. *See* smells.
- Collecting Parameter pattern, 30, 313–319. *See also* Long Method smell.
- Combinatorial Explosion smell, 45
See also Replace Implicit Language with Interpreter.
- Command map, 194
- Command pattern, 30, 191–201
See also Large Class smell
See also Long Method smell
See also Switch Statements smell
- Communicating intention, 58
- Components, supporting multiple versions, 258–268
- Compose Method, 30, 40, 123–128.
See also Long Method smell.
- Composed Method pattern, 30, 40, 121, 123–125. *See also* Long Method smell.
- Composite classes, 215
- Composite pattern
See also Duplicated Code smell
See also Primitive Obsession smell
description, generalization, 224–235
description, simplification, 178–190
encapsulating, 96–113
extracting, 214–223
patterns-based refactoring, 30, 224–235
replacing one/many distinctions, 224–235
- Composite refactoring, 17–20
- Conditional Complexity smell, 41
See also Introduce Null Object
See also Move Embellishment to Decorator

- Conditional Complexity smell, *continued*
 - See also Replace Conditional Logic with Strategy
 - See also Replace State-Altering Conditionals with State
 - Conditional dispatcher, 191–201
 - Conditional logic
 - dispatching requests, 191–201
 - executing actions, 191–201
 - refactoring, 41, 44, 166–177
 - Constants, type-unsafe, 288
 - Constructors
 - assigning values to fields, 346–348
 - catch-all, 341
 - chaining, 340–342
 - communicating intention, 58
 - dead, 58
 - extracting parameters, 346–348
 - naming, 57, 58
 - refactoring, 57–67
 - uncalled code, 58
 - Context classes, 168
 - Continuous refactoring, 5
 - Creation Method pattern
 - description, 57–67
 - Extract Factory, 66–67
 - Parameterized Creation Methods pattern, 65–66
 - patterns-based refactoring, 30
 - vs. Factory Method, 59
 - Creation refactorings
 - Builder pattern
 - description, 96–113
 - encapsulating Composites, 96–113
 - patterns-based refactoring, 30
 - performance improvement, 109–111
 - Schema-Based Builder, 111–113
 - Creation Method pattern
 - description, 57–67
 - Extract Factory, 66–67
 - Parameterized Creation Methods pattern, 65–66
 - patterns-based refactoring, 30
 - vs. Factory Method, 59
 - Encapsulate Classes with Factory, 80–87
 - Encapsulate Composite with Builder, 18, 96–113
 - Factory Method pattern
 - description, 88–95
 - patterns-based refactoring, 31
 - vs. Creation Method, 59
 - Factory pattern
 - description, Encapsulate Classes with Factory, 80–87
 - description, Move Creation Knowledge to Factory, 68–79
 - patterns-based refactoring, 31
 - Inline Singleton, 114–120
 - Introduce Polymorphic Creation with Factory Method, 88–95
 - Move Creation Knowledge to Factory, 68–79
 - Replace Constructors with Creation Method, 57–67
 - Creation sprawl, 69
 - Cunningham, Ward
 - acknowledgments, xxv
 - continuous refactoring, 5
 - design debt, 16
 - human-readable code, 12–13
 - on Singletons, 115
 - TDD (test-driven development), 5
- D**
- Data sprawl, refactoring, 43, 68–79
 - Dead constructors, 58

- Declaration of Independence, rewording, 11–12
- Decorator pattern. *See also*
 - Conditional Complexity smell;
 - Primitive Obsession smell.
- description, 144–165
- patterns-based refactoring, 30
- vs.* Strategy, 147–148
- Delegating classes, 347
- Delegating methods, 133–143
- Design
 - BDUF (big design up-front), 34–35
 - evolutionary, 8, 16–17
- Design debt, 15–16
- Design patterns. *See* patterns.
- Design Patterns*, xxi, xxiii, 24
- Design Patterns Workshop, xxvi
- Design problems, identifying. *See* smells.
- Domain-Driven Design*, xxiv, 24
- Double-dispatch classes, 321
- Duplicated Code smell, 39–40
 - See also* Chain Constructors
 - See also* Extract Composite
 - See also* Form Template Method
 - See also* Introduce Null Object
 - See also* Introduce Polymorphic Creation with Factory Method
 - See also* Replace One/Many Distinctions with Composite
 - See also* Unify Interfaces with Adapter
- E
- Encapsulate Classes with Factory, 80–87. *See also* Indecent Exposure smell.
- Encapsulate Composite with Builder, 18, 96–113. *See also* Primitive Obsession smell.
- Encapsulating
 - classes, 80–87
 - Composites, 96–113
- Evans, Eric, xxiv, 24, 210, 228, 271
- Evolutionary design, 8, 16–17
- Example section, 49
- Execution methods, 193
- External accumulation methods, 323, 325, 326–327
- Extract Adapter, 258–268
- Extract Composite, 214–223. *See also* Duplicated Code smell.
- Extract Factory, 66–67
- Extract Parameter, 346–348
- Extract Superclass, 215
- Extreme Programming Explained*, 24
- F
- Facade pattern *vs.* Adapter pattern, 259–260
- Factory, 70–71
- Factory Method pattern. *See also* Duplicated Code smell.
 - Creation Method, 59
 - description, 88–95
 - patterns-based refactoring, 31
- Factory pattern. *See also* Indecent Exposure smell; Solution Sprawl smell.
 - definition, 70–71
 - description, Encapsulate Classes with Factory, 80–87
 - description, Move Creation Knowledge to Factory, 68–79
 - overuse, 71–72
 - patterns-based refactoring, 31
- Fields
 - null, refactoring, 301–309
 - singleton, 298



- Fields, *continued*
 type safety, 286–295
 type-unsafe, 288
- Form Template Method, 205–213.
See also Duplicated Code smell.
- Fowler, Martin
 acknowledgments, xxiv
 continuous refactoring, 5
 Extract Superclass, 215
 human-readable code, 13
 identifying smells, 37
 Introduce Null Object, 303
 low-level refactorings, xx
 maturity of refactorings, 51–52
 meeting Kerievsky, xxiii
 patterns *vs.* refactorings, 7
 refactoring, definition, 9
 refactoring tools, 20
 on Singletons, 116–117
 TDD (test-driven development), 5
- Franklin, Benjamin, 11–12
- G**
- Gamma, Erich
 acknowledgments, xxiii, xxv
 hard-coded notifications, 237
 information accumulation, 314
 JUnit, 33
- Generalization refactorings
 Adapter pattern
 Adapting with Anonymous
 Inner Classes, 258–268
 description, 247–257
 extracting, 258–268
 patterns-based refactoring, 30
vs. Facade pattern, 259–260
 Composite pattern, 30, 224–235
 Extract Adapter, 258–268
 Extract Composite, 214–223
 Form Template Method, 205–213
 Interpreter pattern, 31, 269–284
 Observer pattern, 31, 236–246
 Replace Hard-Coded Notifications
 with Observer, 236–246
 Replace Implicit Language with
 Interpreter, 269–284
 Replace One/Many Distinctions
 with Composite, 224–235
 Template Method pattern, 31,
 205–213
 Unify Interfaces with Adapter,
 247–257
- H**
- Hat-making anecdote, 11–12
 Hello World example, 24–25
 Heterogeneous classes, 320–338
 HTML Parser, 50–51
 Human-readable code, 12–13
- I**
- Identical methods, 207
 Implementing patterns, 26–29
 Implicit tree structures, 178–190
 Indecent Exposure smell, 42–43. *See also*
Encapsulate Classes with
Factory.
 Information accumulation. *See also*
 Accumulation refactorings.
 classes, 320–338
 methods, from heterogeneous
 classes, 320–338
 methods, to local variable,
 313–319
 Information hiding, 42–43
 Inline Singleton, 114–120. *See also*
 Lazy Class smell.
 Instantiation, limiting, 296–300
 Intent section, 6–7
 Intention, communicating, 58



- Interfaces, unifying, 343–345
- Internal accumulation methods, 322, 325, 327–328
- Interpreter pattern
See also Combinatorial Explosion smell
See also Large Class smell
See also Primitive Obsession smell
 description, 269–284
 patterns-based refactoring, 31
- Interpreting simple languages, 269–284
- Introduce Null Object, 301–309. *See also* Conditional Complexity smell; Duplicated Code smell.
- Introduce Polymorphic Creation with Factory Method, 88–95.
See also Duplicated Code smell.
- Iterator pattern, 31
- J**
- Jefferson, Thomas, 11–12
- Johnson, Ralph
 acknowledgments, xxiii, xxv
 refactoring tools, 20
 on Visitor pattern, 321
- JUnit, 33
- Justifying refactoring to management, 15–16
- L**
- Large Class smell, 44
See also Replace Conditional Dispatcher with Command
See also Replace Implicit Language with Interpreter
See also Replace State-Altering Conditionals with State
- Lazy Class smell, 43–44. *See also* Inline Singleton.
- Libraries, supporting multiple versions, 258–268
- Limit Instantiation with Singleton, 31, 296–300
- Loan risk calculator, 51
- Long Method smell, 40–41
See also Compose Method
See also Composed Method pattern
See also Move Accumulation to Collecting Parameter
See also Move Accumulation to Visitor
See also Replace Conditional Dispatcher with Command
See also Replace Conditional Logic with Strategy
- M**
- Many-object methods, 227–228
- Mechanics section, 48–49
- Memory leaks, 237
- Methods. *See also* Generalization refactorings.
 accept, 327
 accumulation, 315, 325–330
 assigning values to fields, 346–348
 delegating to a Strategy object, 133–143
 execution, 193
 external accumulation, 323, 325, 326–327
 extracting parameters, 346–348
 generalizing, 205–213
 identical, 207
 information accumulation. *See* Accumulation refactorings.
 internal accumulation, 322, 325, 327–328
 many-object, 227–228

Methods, *continued*

- mixing variant and invariant behaviors, 205–213
 - one-object, 227–228
 - partially duplicated, 216
 - purely duplicated, 216
 - refactoring, 40–41, 123–128
 - replacing conditional logic, 129–143
 - similar, 207
 - strategizing, 133–143
 - unique, 207
 - visibility, refactoring, 42–43
- Motivation section, 48
- Motivations for refactoring, 10–11
- Move Accumulation to Collecting Parameter, 313–319. *See also* Long Method smell.
- Move Accumulation to Visitor, 320–338. *See also* Switch Statements smell.
- Move Creation Knowledge to Factory, 68–79. *See also* Solution Sprawl smell.
- Move Embellishment to Decorator. *See also* Conditional Complexity smell; Primitive Obsession smell.
- description, 144–165
 - test-driven refactoring, 19
 - vs.* Replace Conditional Logic with Strategy, 147–148
- Multiple instance classes, 298

N

- Name section, 47
- Naming constructors, 57, 58
- New adapter classes, 261
- Nonterminal expressions, 270
- Notifications, 236–246
- Notifier classes, 238
- Null checks, 304

Null fields, refactoring, 301–309

Null Object pattern. *See also*

- Conditional Complexity smell;

- Duplicated Code smell.

- description, 301–309

- patterns-based refactoring, 31

Null objects

- creating, 304

- Introduce Null Object, 301–309.

- See also* Conditional Complexity smell; Duplicated Code smell.

Null variables, refactoring, 301–309

O

Objects

- creating. *See* Creation refactorings.

- information accumulation. *See*

- Accumulation refactorings.

- state transitions, refactoring,

- 166–177

Observer pattern, 31, 236–246

Oddball Solution smell, 45. *See also*

- Unify Interfaces with Adapter.

One/many distinctions, 224–235

One-object methods, 227–228

Opdyke, William, 20, xxiii

Original state field, 168

Overburdened adapters, 261

Over-engineering, 1–2. *See also*

- Under-engineering.

Overusing patterns, 24–25

P

Papers. *See* Books and publications.

Parameter pattern, 313–319

Parameterized Creation Methods pattern, 65–66

Parnas, David, 42–43

Partially duplicated methods, 216



- Pattern happy, 24
 - A Pattern Language, 23
 - Pattern languages, 23
 - Patterns. *See also specific patterns.*
 - author's comments on, 2–3
 - catalogs of. *See* Catalogs of patterns.
 - and code complexity, 31–32
 - definition, 23–24
 - descriptions of. *See* Intent section.
 - implementing, 26–29
 - individual descriptions. *See* Intent section.
 - overuse of, 24–25
 - purpose. *See* Intent section.
 - refactoring to, towards, and away from, 29–31
 - requisite knowledge, 32–33
 - Structure diagrams, 26–28
 - study sequence, 52–53
 - up-front design, 33–35
 - vs.* refactorings, 7
 - “Patterns & XP,” xxii
 - Patterns of Enterprise Application Architectures*, 24
 - Patterns Reading Group, xxv
 - Polymorphic creation, 88–95
 - Primitive Obsession smell, 41–42
 - See also* Encapsulate Composite with Builder
 - See also* Move Embellishment to Decorator
 - See also* Replace Conditional Logic with Strategy
 - See also* Replace Implicit Language with Interpreter
 - See also* Replace Implicit Tree with Composite
 - See also* Replace State-Altering Conditionals with State
 - See also* Replace Type Code with Class
 - Primitives, refactoring. *See* Primitive Obsession smell.
 - Problems, identifying. *See* Smells.
 - Programs. *See* Classes; Code; Methods.
 - Protecting code. *See* Protection refactorings.
 - Protection refactorings
 - Class pattern, 286–295
 - Introduce Null Object, 301–309
 - Limit Instantiation with Singleton, 31, 296–300
 - Null Object pattern, 31, 301–309
 - Replace Type Code with Class, 286–295
 - Singleton pattern, 31, 296–300
 - Publications. *See* Books and publications.
 - Purely duplicated methods, 216
- R**
- Reasons to refactor. *See* Motivation refactorings.
 - Receiver classes, 238
 - “Red, green, refactor,” 5
 - Refactoring. *See also specific refactorings.*
 - automatic, 20–21
 - composite, 17–20
 - as continuous process, 4–6, 13–14
 - definition, 9
 - evolutionary design, 8, 16–17
 - formats, 47–49
 - justifying to management, 15–16
 - maturity, 51–52
 - motivations for, 10–11
 - overview, 9
 - reasons for. *See* Motivation refactorings.
 - required background, xx–xxi



- Refactoring, *continued*
in small steps, 14–15
study sequence, 52–53
test-driven, 17–19
to, towards, and away from
patterns, 29–31
tools, history of, 20–21
tools, JUnit, 33
vs. patterns, 7
Refactoring, xx, xxi, xxiii
Refactoring@yahoo.com,
xxvi
- Replace Conditional Dispatcher
with Command, 191–201
See also Large Class smell
See also Long Method smell
See also Switch Statements smell
- Replace Conditional Logic with
Strategy, 129–143, 147–148
See also Conditional Complexity
smell
See also Long Method smell
See also Primitive Obsession smell
- Replace Constructors with Creation
Method, 57–67
- Replace Hard-Coded Notifications
with Observer, 236–246
- Replace Implicit Language with
Interpreter, 269–284
See also Combinatorial Explosion
smell
See also Large Class smell
See also Primitive Obsession smell
- Replace Implicit Tree with
Composite, 18–19, 178–190.
See also Primitive Obsession
smell.
- Replace One/Many Distinctions
with Composite, 224–235. *See
also* Duplicated Code smell.
- Replace State-Altering Conditionals
with State, 166–177
See also Conditional Complexity
smell
See also Large Class smell
See also Primitive Obsession smell
- Replace Type Code with Class,
286–295. *See also* Primitive
Obsession smell.
- Request handling, 191–201
- Roberts, Don, 20, xxiii, xxiv
- S**
- Schema-Based Builder, 111–113
- Shotgun Surgery smell, 43
- Silicon Valley Patterns Group
(SVPG), xxiv–xxv
- Similar methods, 207
- Simple and Direct*, 12
- Simplification refactorings
Command pattern, 30,
191–201
Composed Method, 30,
123–128
Composite pattern, 30,
178–190
Decorator pattern, 30, 144–165
Move Embellishment to
Decorator, 19, 144–165
Replace Conditional Dispatcher
with Command, 191–201
Replace Conditional Logic with
Strategy, 129–143, 147–148
Replace Implicit Tree with Com-
posite, 18–19, 178–190
Replace State-Altering Condition-
als with State, 166–177
State pattern, 31, 166–177
Strategy pattern, 31, 129–143,
147–148

- Singleton fields, 298
- Singleton pattern
 - description, 296–300
 - Inline Singleton, 114–120
 - limiting instantiation, 296–300
 - patterns-based refactoring, 31
- Singletonitis, 296–297
- Singletons
 - global access point. *See* Inline Singleton.
 - refactoring, 43–44
- Sketches, 47
- Smalltalk Best Practices Patterns*, 313
- Smells
 - Alternative Classes with Different Interfaces, 43. *See also* Unify Interfaces with Adapter.
 - Combinatorial Explosion, 45. *See also* Replace Implicit Language with Interpreter.
 - Conditional Complexity, 41
 - See also* Introduce Null Object
 - See also* Move Embellishment to Decorator
 - See also* Replace Conditional Logic with Strategy
 - See also* Replace State-Altering Conditionals with State
 - Duplicated Code, 39–40
 - See also* Chain Constructors
 - See also* Combinatorial Explosion smell
 - See also* Extract Composite
 - See also* Form Template Method
 - See also* Introduce Null Object
 - See also* Introduce Polymorphic Creation with Factory Method
 - See also* Oddball Solution smell
 - See also* Replace One/Many Distinctions with Composite
 - See also* Unify Interfaces with Adapter
- Indecent Exposure, 42–43. *See also* Encapsulate Classes with Factory.
- Large Class, 44
 - See also* Replace Conditional Dispatcher with Command
 - See also* Replace Implicit Language with Interpreter
 - See also* Replace State-Altering Conditionals with State
- Lazy Class, 43–44. *See also* Inline Singleton.
- Long Method, 40–41
 - See also* Compose Method
 - See also* Composed Method pattern
 - See also* Move Accumulation to Collecting Parameter
 - See also* Replace Conditional Dispatcher with Command
 - See also* Replace Conditional Logic with Strategy
- most common problems, 37
- Oddball Solution, 45. *See also* Unify Interfaces with Adapter.
- Primitive Obsession, 41–42
 - See also* Encapsulate Composite with Builder
 - See also* Move Embellishment to Decorator
 - See also* Replace Conditional Logic with Strategy
 - See also* Replace Implicit Language with Interpreter
 - See also* Replace Implicit Tree with Composite

Smells, *continued*

- See also* Replace State-Altering Conditionals with State
- See also* Replace Type Code with Class
- recommended refactorings, 38–439
- Shotgun Surgery, 43
- Solution Sprawl, 43. *See also* Move Creation Knowledge to Factory.
- Switch Statements, 44. *See also* Move Accumulation to Visitor; Replace Conditional Dispatcher with Command.
- Solution Sprawl smell, 43. *See also* Move Creation Knowledge to Factory.
- Source code. *See* code.
- Sprawl, refactoring, 43, 68–79
- State pattern, 31, 166–177
 - See also* Conditional Complexity smell
 - See also* Large Class smell
 - See also* Primitive Obsession smell
- State superclass, 168
- State transitions, refactoring, 166–177
- State-altering logic, refactoring, 166–177
- Strategizing methods, 133–143
- Strategy pattern
 - See also* Conditional Complexity smell
 - See also* Long Method smell
 - See also* Primitive Obsession smell
- description, 129–143
- patterns-based refactoring, 31
- vs.* Decorator, 147–148

- Structure diagrams, 26–28
- Subclasses. *See also* Classes.
 - extracting common features, 214–223
 - hard-coded notifications, 236–246
 - implementing the same
 - Composite, 214–223
 - mixing variant and invariant behaviors, 205–213
- Subject superclasses, 238
- Substitute Algorithm, 18
- Summary section, 47–48
- Superclass pattern, 215
- Switch Statements smell, 44. *See also* Move Accumulation to Visitor; Replace Conditional Dispatcher with Command.

T

- TDD (test-driven development), 4–6
- Template Method pattern, 31, 205–213. *See also* Duplicated Code smell.
- Terminal expressions, 270
- Test-Driven Development*, 6
- Test-Driven Development: A Practical Guide*, 6
- Test-driven development (TDD), 4–6
- Test-driven refactoring, 17–19
- Testing & Refactoring Workshop, xxvi
- A Timeless Way of Building*, 23
- Tiscioni, Jason, 24–25
- Tools for refactoring
 - history of, 20–21
 - JUnit, 33
- Tree structures, implicit, 178–190
- Type safety, 286–295
- Type-unsafe constants and fields, 288

U

UML Distilled, xxi

Under-engineering, 3–4. *See also*
Over-engineering.

Unify Interfaces, 343–345

Unify Interfaces with Adapter
See also Alternative Classes with
Different Interfaces smell
See also Duplicated Code smell
See also Oddball Solution smell
description, 247–257

Unifying interfaces, 247–257,
343–345

See also Alternative Classes with
Different Interfaces smell
See also Duplicated Code smell
See also Oddball Solution smell

Unique methods, 207

Up-front design, 33–35

Utilities for refactoring

Chain Constructors, 340–342

Extract Parameter, 346–348

Unify Interfaces, 343–345

See also Alternative Classes with
Different Interfaces smell
See also Duplicated Code smell
See also Oddball Solution smell
See also Unify Interfaces with
Adapter

V

Variables, null, 301–309

Variations section, 49

Versions, supporting multiple,
258–268

Visibility, refactoring, 42–43

Visitee classes, 327

Visitor pattern, 31, 320–338. *See*
also Switch Statements smell.

Vlissides, John, 28, xxiii, xxv

W

Weinberg, Jerry, 33

Woolf, Bobby, xxi, 31–32, 34, 285,
302, 353

X

XML builders, 50